

Le langage SQL

**La manipulation des objets
de la base de données**

C. ROELS

I. Les instructions de mise à jour

**Ajout d'une ligne dans une table
(INSERT)**

**Mise à jour d'une ligne dans une table
(UPDATE)**

**Suppression d'une ligne dans une table
(DELETE)**

**La validation / invalidation d'une transaction
(COMMIT / ROLLBACK)**

**La réservation des lignes dans une table
(LOCK TABLE)**

I.1 Ajout d'une ligne dans une table

```
INSERT INTO nomtable  
VALUES (val1, val2, val3, val4);
```

Si une valeur est donné pour
chaque colonne de la table

```
INSERT INTO nomtable (col1, col2, col3 )  
VALUES (val1, val2, val3 );
```

Indication des colonnes pour
lesquelles une valeur est donnée.

```
INSERT INTO nomtable (col1, col2, col3 )
```

```
Select ....
```

```
From ...
```

```
Where ... ;
```

Insert de valeurs retrouvées avec un SELECT
sur d'autres tables.

Exemples :

a) `INSERT INTO service`

```
VALUES ('FORM', 'Formation', 500, 0, 0, 'F5060');
```

b) `INSERT INTO employe (no_emp, nom_emp, prenom, servis, chef, titre)`

```
VALUES ('409', 'Mendes', 'Mario', 'COMP', 400, 'Comptable');
```

ATTENTION :

les colonnes définies en NOT NULL doivent recevoir une valeur !

c) `INSERT INTO table_salemp`

```
SELECT no_emp, nom_emp, prenom, salair  
FROM employe;
```

I.2 Mise à jour d'une ligne dans une table

```
UPDATE table  
SET      col1 = expr1 , col2 = expr2 , ...  
WHERE    ... ;
```

```
UPDATE table /  
SET      col1 = (select ... from ... where ....)  
WHERE   ....;
```

Mise à jour avec une valeur retrouvée avec un
SELECT sur d'autres tables.

- a) UPDATE employe
SET titre =
'assistante'
WHERE titre = 'secret.' ;

b) UPDATE employe
SET titre = 'ing.comm'
, salair = salair + 1000 , no_tel = 9923
WHERE no_emp = 315 ;

c) UPDATE employe A
SET salair = (SELECT 1.1 *
avg(salair)

FROM employe
WHERE servis = 'prod')
WHERE servis = 'prod' ;

(pour tous les employés du service ‘prod’ : salaire = salaire moyen du service + 10 %)

I.3 Suppression d'une ligne dans une table

DELETE FROM nomtable ;

Suppression de toutes les lignes de la table.

DELETE FROM nomtable

WHERE ... ;

Suppression des lignes
de la table qui
répondent à la condition
du where.

Exemples :

- a) DELETE FROM employe;

b) DELETE FROM employe
WHERE nom_emp = 'MICHAUD';

I.4 La validation / invalidation d'une transaction

COMMIT

Rend permanent toute mise à jour réalisée dans la base de données.

Supprime les "points d'arrêts" qui avaient été définis dans la transaction.

Termine la transaction courante.

ROLLBACK

Termine la transaction courante.

Fait un **undo** de toute mise à jour réalisée dans la base de données.

Supprime les "points d'arrêts" définis dans la transaction.

Exemple COMMIT :

```
INSERT      INTO      employe
VALUES      ( 1234 , ... );
COMMIT ;
```

Exemple ROLLBACK :

```
INSERT INTO ..... ;
SELECT ..... ;
UPDATE ..... ;
ROLLBACK ;
```

I.4 La validation / invalidation d'une transaction (suite)

ROLLBACK to *nom_point_arrest* ;

Un ROLLBACK avec indication d'un point d'arrêt :

Retourne au 'point d'arrêt' spécifié.

Retient le point d'arrêt spécifié, mais supprime ceux définies après.

Exemple ROLLBACK avec point d'arrêt :

Retourne au 'point d'arrêt' spécifié

Retient le point d'arrêt spécifié, mais supprime ceux définies après.

```
INSERT INTO .....;  
savepoint spy  
UPDATE .....;  
savepoint spy2  
DELETE FROM .....;  
SELECT .....;  
ROLLBACK TO SAVEPOINT spy;
```

Note :

Le nombre maximum de savepoints pour une transaction = 5

(peut être modifié dans le fichier init.ora , limité à 255).

I.4 La validation /invalidation d'une transaction (suite)

SET TRANSACTION READ ONLY

Les sélections ne voient que les mises à jour réalisées dans la base de données avant le début de la transaction.

Utile si beaucoup de mises à jour sur des tables qu'on visualise, par les autres utilisateurs !

Ne permet pas d'insert, update, delete dans la base de données.

Se termine au commit ou rollback ou à une commande D.D.L.

I.5 La réservation des lignes dans une table

LOCK TABLE *nom_table1, nom_table2,*

IN *nom_mode* MODE [NOWAIT] ;

Pas d'attente si table déjà
bloquée par quelqu'un
d'autre

ROW SHARE
ROW EXCLUSIVE
SHARE UPDATE
SHARE ROW EXCLUSIVE
EXCLUSIVE

Exclusive	Permet la sélection, pas les autres activités
Share	Permet des sélections concurrentes mais pas d'update
Row share	Accès concurrents permis. Evite le blocage de toute la table pour accès exclusif
Row exclusive	Evite le blocage en share Automatique pour update, insert, delete
Share row exclusive	Sélections dans toute la table Les autres utilisateurs peuvent faire des select, mais ne peuvent pas bloquer la table (même pas en share) ou faire des update

II. L'instruction de consultation

II.1 Les opérations : LA PROJECTION

Permet de spécifier quels **attributs**
(colonnes) on veut retrouver dans la table

II. L'instruction de consultation

II.2 Les opérations : LA RESTRICTION

Permet de spécifier quelles **lignes** on veut retrouver dans la table

II. L'instruction de consultation

II.3 Les liens entre tables

EMPLOYE

NoEmp	Nom	Prenom	CS
101	DUPOND	Jean	100
202	DURAND	Paul	200

SERVICE

CS	Nom Service
100	Commercial
200	Production
300	Comptabilité

Recherche dans les 2 tables,
sans jointure

Recherche dans les 2 tables,
avec jointure

Chaque employé est affecté à un service.

Pour indiquer qu'un employé est affecté à un service, on enrichit la table EMPLOYE par le code service.

C'est la liaison par répétition de valeur .

II. L'instruction de consultation

II.4 Syntaxe générale

```
SELECT      nom-de-colonne ou liste-de-colonne ou *
FROM        nom-de-table ou liste-de-noms-de-tables
Where       condition-de-selection
GROUP BY    ...
HAVING      ...
ORDER BY    ... ;
```

Projection

Restriction

Tri du résultat

II.5 Sélection de la table entière avec projection totale

EXEMPLE :

Afficher la liste complète de la table des services.

ORDRE SQL :

SELECT * FROM SERVICE ;

Sans clause WHERE :
Aucune restriction sur les
lignes à projeter.

Projection de toutes les colonnes de la table spécifiée.

Résultat de l'exemple :

Servis	Nom_serv	No_resp	No_secr	Cod_courrier
COMM	Commercial	100	110	X5021
COMP	Comptabilité	400	410	C3080
PROD	Production	200	210	P2050
PERS	Personnel	300	310	R3075
MARK	Marketing			M5040

II.5 Sélection de la table entière avec projection partielle

EXEMPLE :

Lister les noms, les numéros de matricule et les codes service de tous les employés.

ORDRE SQL :

```
SELECT NOM_EMP , NO_EMP , SERVIS  
FROM EMPLOYE
```

Projection des colonnes spécifiées.

Résultat de l'exemple :

Nom_emp	No_emp	Servis
AGAUT	213	PROD
ARDOINGT	216	PROD
BALLANT	212	PROD
BARBIET	300	PERS
BEZAUX	120	COMM
BOULUGE	122	COMM
CANNET	310	PERS
CHARTIER	411	COMP
DECROIX	412	COMP
DELOIN	200	PROD

Etc.

II.5 Sélection de la table entière avec tri du résultat

EXEMPLE :

Lister les noms, prénoms et titres de tous les employés, triés par ordre alphabétique des titres.

ORDRE SQL :

```
SELECT      NOM_EMP , PRENOM , TITRE  
FROM        EMPLOYE  
ORDER BY    TITRE ;
```

La colonne peut être suivi de DESC ou ASC (par défaut).

Il est possible de spécifier plusieurs colonnes de tri.
Chaque colonne spécifiée peut être suivi de DESC ou ASC.

Exemple :

```
Select servis, nom_emp, prenom, titre, salaire  
From employe  
Order by servis asc, titre asc, salaire desc;
```

Ici le tri sera fait sur le service et le titre de façon ascendante, puis sur le salaire de façon descendante.

II.5 Sélection avec élimination des doublons

EXEMPLE :

Lister les codes des services dans lesquels travaille au moins un employé.

ORDRE SQL :

```
SELECT      DISTINCT  SERVIS  
FROM        EMPLOYE  
ORDER BY    SERVIS ;
```

Sans l'utilisation du DISTINCT, la requête SQL afficherait autant de lignes qu'il y ait d'employés.

On afficherait donc plusieurs fois les mêmes codes service!

REMARQUE :

Le DISTINCT s'applique à TOUTE la projection!

Exemple :

```
Select      DISTINCT  SERVIS, SALAIRE  
From       employe  
Order by   Servis, salaire;
```

Ici, le même service sera affiché plusieurs fois : autant de fois qu'il y ait de salaires différents dans le service.

II.6 Restriction de lignes

- Opérateurs de comparaison

=	!=	>	>=	<	<=
---	----	---	----	---	----

EXEMPLE :

Liste des caractéristiques de l'employé qui porte le numéro 411.

ORDRE SQL :

```
SELECT * FROM EMPLOYE  
WHERE NO_EMP = 411;
```

II.6 Restriction de lignes

- Opérateurs logiques

EXEMPLES :

```
SELECT * FROM EMPLOYE  
WHERE SERVIS = 'COMM'  
AND SALAIR < 8000 ;
```

```
SELECT * FROM EMPLOYE  
WHERE NOT( SERVIS = 'COMM'  
        AND  
        SALAIR < 8000 );
```

```
SELECT * FROM EMPLOYE  
WHERE SERVIS = 'COMM'  
OR SALAIR < 8000 ;
```

II.6 Restriction de lignes

- Opérateur IN

EXEMPLE :

Lister les noms et les services respectifs des employés des services PERSONNEL et COMPTABLE, triés par service et par nom.

ORDRE SQL :

```
SELECT      NOM_EMP , SERVIS  
FROM        EMPLOYE  
WHERE       SERVIS IN ( 'PERS' , 'COMP' )  
ORDER       BY   SERVIS , NOM_EMP ;
```

II.6 Restriction de lignes

- Opérateur BETWEEN

EXEMPLE :

Lister les numéros et noms des employés dont le numéro est compris entre 200 et 299 (bornes incluses).

ORDRE SQL :

```
SELECT      NO_EMP , NOM_EMP  
FROM        EMPLOYE  
WHERE       NO_EMP BETWEEN 200 AND 299 ;
```

II.6 Restriction de lignes

- Opérateur LIKE

EXEMPLES :

```
SELECT      NOM_EMP, SERVIS, TITRE  
FROM        EMPLOYE  
WHERE       TITRE LIKE 'ING%'
```

```
SELECT      NOM_EMP, SERVIS, TITRE  
FROM        EMPLOYE  
WHERE       NOM_EMP LIKE '_A%'
```

```
SELECT      NOM_EMP, SERVIS, TITRE  
FROM        EMPLOYE  
WHERE       NOM_EMP LIKE 'M_____';
```

L'opérateur LIKE utilise des caractères 'joker' :

- le caractère _ (souligné) remplace 1 caractère,
- Le caractère % remplace une suite de caractères.

II.7 Les 5 fonctions de groupe

MIN	MAX	SUM	AVG	COUNT
-----	-----	-----	-----	-------

```
SELECT      MIN(salair), MAX(salair),  
           AVG(salair), SUM(salair)  
FROM        EMPLOYE  
WHERE       TITRE LIKE 'ING%' ;
```

```
SELECT      COUNT (*)  
FROM        EMPLOYE  
WHERE       SERVIS = 'COMP' ;
```

II.8 Les calculs

```
SELECT      MAX(salair) , MIN(salair) ,
            MAX(salair) - MIN(salair)
FROM        EMPLOYE ;
```

```
SELECT      NO_EMP, SERVIS,
            SALAIR, 1.05 * SALAIR
FROM        EMPLOYE
WHERE       SERVIS = 'COMP' ;
```

II.9 Sélections multi-tables

- jointure sans restriction

UNE SELECTION DANS PLUSIEURS TABLES
= une jointure

UNE JOINTURE SANS RESTRICTION
= le produit cartésien des tables.

EXEMPLE :

```
SELECT      NOM_EMP , PRENOM , NOM_SERV  
FROM        EMPLOYE , SERVICE ;
```

* nombre de lignes important ($M \times N$).
* certaines lignes n'ont pas de sens.

II.9 Sélections multi-tables

- jointure avec restriction

UNE JOINTURE AVEC RESTRICTION
le produit cartésien des tables + restriction

EXEMPLE :

```
SELECT      NOM_EMP , PRENOM , NOM_SERV  
FROM        EMPLOYE , SERVICE  
WHERE       EMPLOYE.SERVIS = SERVICE.SERVIS;
```

Lorsqu'une colonne est identique dans plusieurs tables,
il faut la qualifier par le forme :

NOM_de_TABLE.NOM_de_COLONNE

Afin de faciliter l'écriture des conditions de jointure, il est possible de renommer les tables dans la clause FROM et d'utiliser ensuite ces nouveaux noms dans la condition de jointure.

Exemple :

```
SELECT      NOM_EMP , PRENOM , NOM_SERV  
FROM        EMPLOYE e , SERVICE s  
WHERE       e.SERVIS = s.SERVIS;
```

Si plus de 2 tables sont utilisées dans la requête, il faut indiquer toutes les conditions de jointure nécessaires.

II.9 Sélections multi-tables

- jointure avec restriction et +

Jointure avec restriction par égalité
et condition logique

```
SELECT NOM_EMP, PRENOM, NOM_SERV,  
      CD_COURRIER  
  FROM EMPLOYE e, SERVICE s  
 WHERE e.SERVIS = s.SERVIS  
   AND e.SERVIS = 'COMM';
```

```
SELECT NOM_EMP , PRENOM , SALAIR ,  
      TITRE , NOM_SERV  
  FROM EMPLOYE , SERVICE  
 WHERE e.SERVIS = s.SERVIS  
   AND TITRE LIKE 'ING%'  
   AND SALAIRE >= 9000 ;
```

II.9 Sélections multi-tables - jointure externe

```
Select nom_serv, count(*)  
From service s, employe e  
Where s.servis = e.servis;
```

Affiche les services avec le
nombre d'employés
(uniquement pour les services
contenant des employés)

```
Select nom_serv, count(*)  
From service s, employe e  
Where s.servis = e.servis(+);
```

Affichera également les services
n'ayant pas d'employés

II.10 Les groupements

```
SELECT      SERVIS , AVG (SALAIRE)
FROM        EMPLOYE
GROUP      BY SERVIS ;
```

```
SELECT      SERVIS , MAX (SALAIRE)
FROM        EMPLOYE
GROUP      BY SERVIS ;
```

Règle à appliquer :

Si un SELECT contient 1 des 5 fonctions de groupe (MIN, MAX, SUM, AVG, COUNT) à côté de colonnes 'normales' ,

Il faut mettre toutes les colonnes 'normales' dans une clause GROUP BY.

Le GROUP BY indique que la fonction doit être exécutée sur un groupe de lignes.

Chaque groupe est constitué de lignes ayant des valeurs identiques dans les colonnes 'normales'.

Le SGBD exécute la requête en 2 temps :

- 1) Select des colonnes normales et de la colonne utilisée dans la fonction de groupe. Le résultat est mémorisé pour la deuxième étape
- 2) Application de la fonction de groupe sur chaque groupe de lignes.

II.10 Les groupements

- Cas particulier : la fonction COUNT

```
SELECT      SERVIS , BUREAU , COUNT (*)  
FROM        EMPLOYE  
GROUP BY   SERVIS , BUREAU ;
```

```
SELECT      SERVIS, COUNT( distinct bureau )  
FROM        EMPLOYE  
GROUP BY   SERVIS;
```

La fonction COUNT ne nécessite l'utilisation d'une colonne QUE si on souhaite compter les valeurs distinctes de la colonne.

II.10 Les groupements

- La clause HAVING

La clause HAVING est à la sélection de groupes
ce que la clause WHERE est à la sélection de lignes.

```
SELECT      BUREAU , COUNT(*)  
FROM        EMPLOYE  
GROUP BY   BUREAU  
HAVING     COUNT (*) > 1 ;
```

Toute restriction sur le résultat d'une fonction de groupe doit être spécifiée
dans une clause HAVING,
après la clause GROUP BY.

II.11 Les requêtes imbriquées

Une sélection de sélection =
une sous-requête ou une requête imbriquée

```
SELECT      *  
FROM        EMPLOYE  
WHERE       SALAIRE >  
           ( SELECT AVG(SALAIR)  
             FROM EMPLOYE );
```

1) La sous-requête recherche la
moyenne des salaires des employés

2) La requête globale recherche les informations des
employés ayant un salaire > résultat de la sous-requête

Autre utilisation des sous-requêtes :

optimiser les temps d'accès aux données en évitant les jointures.

```
Select  *  from employe  
Where   servis IN (select servis from service where budget > 10000);
```

```
Select  *  from employe e, service s  
Where   e.servis = s.servis  
And     budget > 10000;
```

La sous-requête doit être entre parenthèses.

Elle a la même forme qu'une requête mais :

- ne peut apparaître que comme second opérande d'une comparaison
- n'admet qu'une colonne dans la clause SELECT,
- ne peut pas avoir de clause ORDER BY.

II.11.1 Les requêtes imbriquées – EXISTS

EXEMPLE :

Noms des services dans lesquels il y a des ingénieurs ?

ORDRE SQL :

```
SELECT      NOM_SERV  
FROM        SERVICE  
WHERE       EXISTS
```

Requête corrélée :
utilisant une jointure avec une
table de la requête globale.

```
(SELECT *  FROM EMPLOYE  
  WHERE     TITRE LIKE 'ING%'  
  AND EMPLOYE.SERVIS = SERVICE.SERVIS)
```

EXISTS : opérateur d'existence dans une clause WHERE, nécessitant une requête imbriquée corrélée.

Autre façon d'écrire la requête :

```
SELECT      NOM_SERV  
FROM        SERVICE  
WHERE       SERVIS IN  
           (SELECT SERVIS FROM EMPLOYE);
```

II.11.2 Les requêtes imbriquées – ALL

EXEMPLE :

Quel service a le plus d'employés et combien ?

ORDRE SQL :

```
SELECT      SERVIS , COUNT(*)
FROM        EMPLOYE
GROUP       BY SERVIS
HAVING     COUNT (*) >= ALL
( SELECT    COUNT(*)
  FROM EMPLOYE
  GROUP BY SERVIS );
```

Même requête qu'en haut, mais projetant uniquement le résultat de la fonction de groupe.

1	2
SELECT SERVIS, COUNT(*) FROM EMPLOYE GROUP BY SERVIS	SELECT COUNT(*) FROM EMPLOYE GROUP BY SERVIS
PROD 6	6
PERS 4	4
COMM 5	5
COMP 5	5

3
HAVING COUNT(*) >= ALL
PROD 6

Chaque ligne du résultat de la requête 1 EST COMPAREE avec chaque ligne du résultat de la requête 2

Une ligne (requête 1) est retenue UNIQUEMENT si COUNT(*) >= à tous (ALL) les résultats de la requête 2

II.12 Les copies de table dans une requête

EXEMPLE :

Afficher le nom et salaire des responsables et de leur secrétaire, côté à côté.

REQUETE SQL :

```
SELECT      s.servis, e1.nom_emp , e1.prenom ,  
                  e2.nom_emp , e2.prenom  
FROM        SERVICE s, EMPLOYE e1, EMPLOYE e2  
WHERE       e1.no_emp    = s.no_resp  
AND         e2.no_emp    = s.no_secr ;
```

Pour 1 ligne de la table SERVICE, on doit retrouver 2 lignes différentes dans la table EMPLOYE :

- 1 ligne pour le responsable de service
- 1 ligne pour la secrétaire du service

Il faut écrire la requête comme si ces 2 ligne devraient être retrouvées dans 2 tables différentes.

Solution :

- indiquer 2 fois la table EMPLOYE dans le FROM avec renommage
- utiliser les renommages dans les conditions de jointure

II.13 Les requêtes imbriquées corrélées

Une sous-requête dont la restriction réfère à la requête principale est appelée une sous-requête corrélée.

La sous-requête n'est donc pas indépendante.

EXEMPLE :

Quels sont les employés qui gagnent plus de la moyenne de leur service ?

ORDRE SQL :

```
SELECT      NOM_EMP , PRENOM ,
            TITRE , SALAIR , SERVIS
  from    employe m
 where   salair >=
        ( SELECT      AVG(SALAR)
          FROM       EMPLOYE
          WHERE      EMPLOYE.SERVIS = M.SERVIS )
 ORDER BY SERVIS ;
```

III. Les opérateurs ensemblistes

III.1 UNION

III.2 INTERSECT

III.3 MINUS

III.1 UNION



Exemple :

Rechercher tous les ingénieurs (tous services confondus), ainsi que les employés du service RECH (recherche).

```
SELECT NOM_EMP , PRENOM , TITRE ,  
       SALAIR , SERVIS  
  FROM EMPLOYE  
 WHERE TITRE LIKE 'ING%'
```

REQ 1 :
Recherche de tous
les ingénieurs

UNION

```
SELECT NOM_EMP , PRENOM , TITRE ,  
       SALAIR , SERVIS  
  FROM EMPLOYE  
 WHERE SERVIS = 'RECH' ;
```

Regroupe les résultats des 2
requêtes

REQ 2 :
Recherche des
employés du
service RECH

Les 2 requêtes dont le résultat seront regroupés, doivent avoir la même structure :

- même nombre de colonnes dans le SELECT
- même format pour les colonnes

III.2 INTERSECT



Exemple :

Rechercher tous les ingénieurs du service RECH (recherche).

```
SELECT NOM_EMP , PRENOM , TITRE ,  
       SALAIR , SERVIS  
  FROM EMPLOYE  
 WHERE TITRE LIKE 'ING%'
```

REQ 1 :
Recherche de tous
les ingénieurs

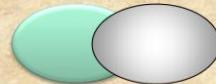
INTERSECT

```
SELECT NOM_EMP , PRENOM , TITRE ,  
       SALAIR , SERVIS  
  FROM EMPLOYE  
 WHERE SERVIS = 'RECH' ;
```

Trouve les enregistrements
communs aux 2 requêtes.

REQ 2 :
Recherche des
employés du
service RECH

III.3 MINUS



Exemple :

Rechercher tous les ingénieurs,
SAUF ceux du service RECH (recherche).

```
SELECT NOM_EMP , PRENOM , TITRE ,  
       SALAIR , SERVIS  
  FROM EMPLOYE  
 WHERE TITRE LIKE 'ING%'  
MINUS
```

REQ 1 :
Recherche de tous
les ingénieurs

Soustrait le résultat de la
REQ2 de la REQ1

```
SELECT NOM_EMP , PRENOM , TITRE ,  
       SALAIR , SERVIS  
  FROM EMPLOYE  
 WHERE TITRE LIKE 'ING%'  
   AND SERVIS = 'RECH' ;
```

REQ 2 :
Recherche des
ingénieurs du
service RECH

ANNEXES

**Recherche dans 2 tables
SANS jointure**

NoEmp	Nom	Prenom	CS	CS	Nom service
101	DUPOND	Jean	100	100	Commercial
101	DUPOND	Jean	100	200	Production
101	DUPOND	Jean	100	300	Comptabilité
202	DURAND	Paul	200	100	Commercial
202	DURAND	Paul	200	200	Production
202	DURAND	Paul	200	300	Comptabilité

[Retour](#)

Tous les résultats n'ont pas une signification intéressante.

Recherche dans 2 tables AVEC jointure

PRODUIT CARTESIEN + RESTRICTION
= OPERATION DE JOINTURE

NoEmp	Nom	Prenom	CS	CS	Nom service
101	DUPOND	Jean	100	100	Commercial
202	DURAND	Paul	200	200	Production

[Retour](#)

- (1) La restriction qui met en œuvre une condition d'égalité est une **EQUI-JOINTURE**.
- (2) La restriction qui met en œuvre une condition autre que l'égalité ($>$, $<$, \geq , ...) est une **THETA-JOINTURE**,
opérateur très puissant pour de nombreuses requêtes.